CLAIMS

What is claimed is:

1.   A method for implementing subroutine calls and returns in a computer system comprising the following steps:

    A) converting a sequence of input language (IL) instructions into a corresponding sequence of output language (OL) instructions;

    B) executing the OL instructions;

    C) for each call to an IL subroutine made from an IL call site in the IL instruction sequence:

        i)   storing a call site IL return address R_{call} on a stack;

        ii)   calculating a first index by evaluating a function with P as an argument, where P is a procedure entry address of the subroutine;

        iii)   storing a corresponding OL return address in a return target cache at a location indicated by the first index;

        iv)   executing an OL subroutine translation of the called IL subroutine;

    D) upon completion of execution of the OL subroutine translation,

        i)   in a launch block of instructions, retrieving an OL target address from the return target cache at the location indicated by a second index; and

        ii)   continuing execution beginning at the OL target address.

2.   A method as in claim 1, further including the following steps:

    determining whether a predicted IL return address R_{pred} is the same as an actual IL return address R_{actual} fetched from the stack and, if it is not, transferring execution to a back-up OL return address recovery module; and

    in the back-up OL return address recovery module, establishing the OL return address using a predetermined, secondary address recovery routine.

1   3.   A method as in claim 2, in which there is a plurality of IL call sites, further

2 including the following steps:

3   translating each IL call site into a corresponding OL call site;

4   generating a confirm block of instructions corresponding to each OL call site;

5   upon execution of any confirm block of instructions:

6   comparing the actual IL return address R_{actual} with the predicted IL return

7 address R_{pred};

8   if R_{actual} is equal to R_{pred}, continuing execution of the OL instructions

9 following the OL call site; and

10   if R_{actual} is not equal to R_{pred}, transferring execution to the back-up return

11 address recovery module.

1   4.   A method as in claim 3, in which only a single scratch register is used for

2 the in the launch and confirmation blocks of instructions.

1   5.   A method as in claim 3, in which:

2   the return target cache is an array having a plurality of elements;

3   the function maps IL return addresses substantially uniformly over the return

4 target cache;

5   equality and inequality between R_{actual} and R_{pred} are defined as a hit and a miss,

6 respectively;

7   further including the following steps:

8   calculating a return success measure as a function of the frequency of

9 occurrence of hits relative to the frequency of occurrence of misses;

10   adjusting the number of elements in the return target cache according to a

11 function of the return success measure.

1   6.   A method as in claim 2, in which the return target cache is an array having

2 a plurality of elements, further including the step of initializing the return target cache by

3 storing in each element a beginning address of the back-up return address recovery

4 module.

1   7.   A method as in claim 1, in which:

2   the return target cache is an array having a plurality of elements; and

3   the function maps IL procedure entry addresses substantially uniformly over the

4   return target cache.

1   8.   A method as in claim 7, in which each of the elements of the return target

2   cache is identified by an array index, and the function extracts a number of bits from the

3   address P.

1   9.   A method as in claim 1, in which the step of calculating the first index k is

2   performed as part of the step of converting the IL call into the corresponding sequence

3   of OL instructions.

1   10.   A method as in claim 1, further comprising binding a translation of a return

2   within the OL subroutine translation to an index in the return target cache.

1   11.   A method as in claim 10, further comprising:

2   setting a specified entry of the return target cache to a predetermined value

3   indicating a lack of binding; and

4   upon sensing attempted access to the specified entry of the return target cache,

5   scanning the return target cache and associating with the current unbound launch block

6   an array index other than the specified index.

1   12.   A method for implementing subroutine calls and returns in a computer

2   system comprising the following steps:

3   A) converting a sequence of input language (IL) instructions of a guest system

4   into a corresponding sequence of output language (OL) instructions of a host system;

5   B) executing the OL instructions in the host system;

6   C) for each call to an IL subroutine made from any of a plurality of IL call sites in

7   the IL instruction sequence:

8   i)   translating each IL call site into a corresponding OL call site;

9   ii)   storing a call site IL return address R_{call} on a stack;

10   iii)   calculating a first index by evaluating a function with P as an

11   argument, where P is a procedure entry address of the subroutine;

12   iv)   storing a corresponding OL return address R' in a return target

13   cache at a location determined by the first index, the return target cache comprising an

14   array of elements;

15   v)   executing an OL subroutine translation of the called IL subroutine;

16   D) upon completion of execution of the OL subroutine translation,

17   i)   retrieving an OL target address from the return target cache at the

18   location indicated by a second index; and

19   ii)   continuing execution beginning at the OL target address.

20   E) generating a confirm block of instructions corresponding to each OL call site

21   and, upon execution of any confirm block of instructions:

22   i) comparing an actual IL return target address R_{actual} fetched from the

23   stack with the predicted IL return address R_{pred};

24   ii) if R_{actual} is equal to R_{pred}, continuing execution of the OL instructions

25   following the OL call site; and

26   iii) if R_{actual} is not equal to R_{pred}, transferring execution to the back-up

27   return address recovery module; and

28   F) in the back-up return address recovery module, determining a correct OL

29   return address.

1   13.   A method as in claim 12, further comprising binding a translation of a

2   return within the OL subroutine translation to an index in the return target cache.

1   14.   A system for implementing subroutine calls and returns in a computer

2   system comprising:

3   A) a host computer system that executes host instructions in an output language

4   OL;

5   B) a guest system that is operatively connected to the host system and that

6   issues a sequence of instructions in an input language (IL) including a call to a

7   subroutine;

8   C) a binary translator converting the sequence of input language (IL) instructions

9   of the guest system into a corresponding sequence of the output language (OL)

10   instructions of the host system and storing the OL instructions in a translation cache,

11   D) the binary translator comprising computer-executable instructions for

12   translating an IL subroutine call and an IL subroutine return into corresponding OL

13   instruction sequences, including a call block and a launch block of OL instructions;

14   E) the call block, upon each call to an IL subroutine from an IL call site in the IL

15   instruction sequence, comprising computer-executable instructions

16   i)   for storing a call site IL return address R_{call} of the call on a stack;

17   ii)   for determining a first index by evaluating a function with P as an

18   argument, where P is a procedure entry address of the subroutine; and

19   iii)   for storing a corresponding OL return address R' in a return target

20   cache at a location determined by the first index;

21   iv)   for transferring execution to the OL subroutine translation of the

22   called IL subroutine;

23   F) the launch block, upon completion of execution of the OL subroutine

24   translation, further comprising computer-executable instructions

25   i)   for popping an actual IL return address R_{actual} from the stack;

26   ii)   for retrieving an OL target address from the return target cache at

27   the location indicated by a second index; and

28   iii)   for continuing execution beginning at the OL target address.

1   15.   A system as in claim 14, in which:

2   there is a plurality of IL call sites;

3   the binary translator comprises computer-executable instructions

4   for translating each IL call site into a corresponding OL call site;

5   for inserting a confirm block of instructions into each OL call site;

6   for comparing R_{actual} with a predicted IL return address R_{pred} corresponding

7 to the current OL call site;

8   for continuing execution of the OL instructions following the OL call site if

9 R_{actual} is equal to R_{pred}; and

10   for transferring execution to the back-up return address recovery module if

11 R_{actual} is not equal to R_{pred}.

1   16.   A system as in claim 14, in which the binary translator comprises further

2 computer-executable instructions for binding a translation of a return within the OL

3 subroutine translation to an index in the return target cache.